Exploratory data analysis (EDA) using python: a tutorial

Presentation · October 2024

DOI: 10.13140/RG.2.2.20542.09284/1

1 author:

Sami D. Alaruri سامي العاروري
Independnet Researcher

EXPLORATORY DATA ANALYSIS (EDA) USING PYTHON: A TUTORIAL

SAMI D. ALARURI

OCT., 2024

AGENDA

- Introduction
- Overview
- Data Inspection & Cleaning Steps
- Graphical EDA
- Conclusions
- References

INTRODUCTION: WHAT IS EDA?

Exploratory data analysis is an analysis technique to analyze and investigate the data set and summarize the main characteristics of the dataset. Main advantage of EDA is providing the data visualization of data after conducting the analysis.

Tukey defined data analysis in 1961 as: "Procedures for analyzing data, techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easier, more precise or more accurate, and all the machinery and results of (mathematical) statistics which apply to analyzing data”.
https://en.wikipedia.org/wiki/Exploratory_data_analysis

OVERVIEW

Obesity is a complex disease involving having too much body fat. Obesity isn't just a cosmetic concern. It's a medical problem that increases the risk of many other diseases and health problems. These can include heart disease, diabetes, high blood pressure, high cholesterol, liver disease, sleep apnea and certain cancers.

There are many reasons why some people have trouble losing weight. Often, obesity results from inherited, physiological and environmental factors, combined with diet, physical activity and exercise choices.

The good news is that even modest weight loss can improve or prevent the health problems associated with obesity. A healthier diet, increased physical activity and behavior changes can help you lose weight. Prescription medicines and weight-loss procedures are other options for treating obesity. In this Exploratory Data Analysis (EDA) using Python we will examine the NObeyesdad (Obesity level of the individual) as a function of several factors (i.e., age, gender, height, alcohol consumption ..etc.).

https://www.mayoclinic.org/diseases-conditions/obesity/symptoms-causes/syc-20375742

DATASET INFORMATION

This dataset include data for the estimation of obesity levels in individuals from the countries of Mexico, Peru and Colombia, based on their eating habits and physical condition. The data contains 17 attributes and 2111 records, the records are labeled with the class variable NObesity (Obesity Level), that allows classification of the data using the values of Insufficient Weight, Normal Weight, Overweight Level I, Overweight Level II, Obesity Type I, Obesity Type II and Obesity Type III. 77% of the data was generated synthetically using the Weka tool and the SMOTE filter, 23% of the data was collected directly from users through a web platform.

https://archive.ics.uci.edu/dataset/544/estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition

DATA INSPECTION & CLEANING STEPS

1. Dataset dimensions
2. Titles of columns
3. Data types
4. Missing values
5. Nulls in the dataset (not required for this dataset)
6. Duplicate rows
7. NaN values
8. Infinity values
9. Outliers detection
10. Encode categorical features

LOADING PYTHON LIBRARIES

```
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder, StandardScaler,LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
warnings.filterwarnings("ignore")
```

pandas (https://pandas.pydata.org)

numpy (https://numpy.org)

matplotlib (https://matplotlib.org)

seaborn (https://seaborn.pydata.org)

sklearn (https://scikit-learn.org/stable)

DATA SET ATTRIBUTES

1. Age: The age of the individual.
2. Gender: The gender of the individual (e.g., Male, Female).
3. Height: The height of the individual in meters.
4. Weight: The weight of the individual in kilograms.
5. CALC: Unknown column. You might need to check the data source or documentation to understand what this column represents.
6. FAVC: Whether the individual frequently consumes high caloric food (e.g., yes, no).
7. FCVC: Frequency of consumption of vegetables (numeric scale).
8. NCP: Number of main meals per day (numeric scale).
9. SCC: Squamous cell carcinoma
10. SMOKE: Whether the individual smokes (e.g., yes, no).
11. CH2O: Consumption of water daily (numeric scale).
12. family_history_with_overweight: Whether the individual has a family history of overweight (e.g., yes, no).
13. FAF: Physical activity frequency (numeric scale).
14. TUE: Time using technology devices (numeric scale).
15. CAEC: Unknown column. You might need to check the data source or documentation to understand what this column represents.
16. MTRANS: Mode of transportation (e.g., Public Transportation, Walking).
17. NObeyesdad: Obesity level of the individual (e.g., Normal_Weight, Overweight_Level_I).
    Note: The data set contains a mixture of numeric and categorical data.

PANDAS FUNCTIONS FOR DATA INSPECTION

- df.head()/df.tail()
- df.sample()
- df.info()
- df.columns
- df.describe()
- df.shape
- Df.count()

https://www.geeksforgeeks.org/pandas-functions-in-python/

USING PANDAS FOR LOADING THE DATA FILE & VIEWING THE FIRST 5 ROWS

```
data = pd.read_csv("/content/sample_data/ObesityDataSet_raw_and_data.csv") # Load data set using pandas
data.head(5)
```

	Age	Gender	Height	Weight	CALC	FAVC	FCVC	NCP	SCC	SMOKE	CH2O	family_history_with_overweight	FAF	TUE	(
0	21.0	Female	1.62	64.0	no	no	2.0	3.0	no	no	2.0	yes	0.0	1.0	Some
1	21.0	Female	1.52	56.0	Sometimes	no	3.0	3.0	yes	yes	3.0	yes	3.0	0.0	Some
2	23.0	Male	1.80	77.0	Frequently	no	2.0	3.0	no	no	2.0	yes	2.0	1.0	Some
3	27.0	Male	1.80	87.0	Frequently	no	3.0	3.0	no	no	2.0	no	2.0	0.0	Some
4	22.0	Male	1.78	89.8	Sometimes	no	2.0	1.0	no	no	2.0	no	0.0	0.0	Some

Dataset source: UC Irvine-Machine Learning Repository

https://archive.ics.uci.edu/dataset/544/estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition

EXAMINING 5 ROWS IN RANDOM & AT THE END OF THE DATASET

```python
data.sample(5)
```

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	
1679	Male	31.761799	1.751688	119.205308	yes	yes	2.149610	3.000000	Sometimes	no	2.133876	no	0.393452	0.3
752	Male	21.142432	1.855353	86.413388	yes	yes	2.000000	3.000000	Sometimes	no	1.345298	no	1.097905	1.0
1546	Male	25.298400	1.827279	120.996074	yes	yes	3.000000	3.000000	Sometimes	no	3.000000	no	1.110215	0.3
1741	Male	28.255199	1.816547	120.699119	yes	yes	2.997951	3.000000	Sometimes	no	2.715856	no	0.739881	0.9
1790	Male	23.147644	1.815514	120.337664	yes	yes	2.996717	2.791366	Sometimes	no	2.626309	no	1.194898	0.0

```python
data.tail(5)
```

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE
2106	Female	20.976842	1.710730	131.408528	yes	yes	3.0	3.0	Sometimes	no	1.728139	no	1.676269	0.906247
2107	Female	21.982942	1.748584	133.742943	yes	yes	3.0	3.0	Sometimes	no	2.005130	no	1.341390	0.599270
2108	Female	22.524036	1.752206	133.689352	yes	yes	3.0	3.0	Sometimes	no	2.054193	no	1.414209	0.646288
2109	Female	24.361936	1.739450	133.346641	yes	yes	3.0	3.0	Sometimes	no	2.852339	no	1.139107	0.586035
2110	Female	23.664709	1.738836	133.472641	yes	yes	3.0	3.0	Sometimes	no	2.863513	no	1.026452	0.714137

USING PANDA FUNCTIONS FOR VIEWING THE DATA

```
In [136]: data.columns

Out[136]: Index(['Age', 'Gender', 'Height', 'Weight', 'CALC', 'FAVC', 'FCVC', 'NCP',
                 'SCC', 'SMOKE', 'CH2O', 'family_history_with_overweight', 'FAF', 'TUE',
                 'CAEC', 'MTRANS', 'NObeyesdad'],
                dtype='object')

In [138]: data.shape # Fetch the data set dimensions before cleaning

Out[138]: (2111, 17)
```

Data set size:
2111 rows x 17 columns

```
data.info() # check data type in each column

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2111 entries, 0 to 2110
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Age                             2111 non-null   float64
 1   Gender                          2111 non-null   object
 2   Height                          2111 non-null   float64
 3   Weight                          2111 non-null   float64
 4   CALC                            2111 non-null   object
 5   FAVC                            2111 non-null   object
 6   FCVC                            2111 non-null   float64
 7   NCP                             2111 non-null   float64
 8   SCC                             2111 non-null   object
 9   SMOKE                           2111 non-null   object
 10  CH2O                            2111 non-null   float64
 11  family_history_with_overweight  2111 non-null   object
 12  FAF                             2111 non-null   float64
 13  TUE                             2111 non-null   float64
 14  CAEC                            2111 non-null   object
 15  MTRANS                          2111 non-null   object
 16  NObeyesdad                      2111 non-null   object
dtypes: float64(8), object(9)
memory usage: 280.5+ KB
```

DATA STATISTICS & MISSING VALUES CHECK

```
data.describe()
```

	Age	Height	Weight	FCVC	NCP	CH2O	FAF	TUE
count	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000
mean	24.312600	1.701677	86.586058	2.419043	2.685628	2.008011	1.010298	0.657866
std	6.345968	0.093305	26.191172	0.533927	0.778039	0.612953	0.850592	0.608927
min	14.000000	1.450000	39.000000	1.000000	1.000000	1.000000	0.000000	0.000000
25%	19.947192	1.630000	65.473343	2.000000	2.658738	1.584812	0.124505	0.000000
50%	22.777890	1.700499	83.000000	2.385502	3.000000	2.000000	1.000000	0.625350
75%	26.000000	1.768464	107.430682	3.000000	3.000000	2.477420	1.666678	1.000000
max	61.000000	1.980000	173.000000	3.000000	4.000000	3.000000	3.000000	2.000000

Dataset Statistics Summary

```
# Check for missing values in the DataFrame
missing_values = data.isnull().sum()
print("Missing Values:")
print(missing_values)
```

```
Missing Values:
Age                               0
Gender                            0
Height                            0
Weight                            0
CALC                              0
FAVC                              0
FCVC                              0
NCP                               0
SCC                               0
SMOKE                             0
CH2O                              0
family_history_with_overweight    0
FAF                               0
TUE                               0
CAEC                              0
MTRANS                            0
NObeyesdad                        0
dtype: int64
```

As seen, there are no missing values in this dataset

Missing Values Check
No missing values

INSPECTING THE DATASET FOR DUPLICATE ROWS & DROPPING THE DUPLICATE ROWS

```
# Check for duplicate rows in the data set
duplicate_rows = data.duplicated().sum()
print("Duplicate Rows:")
print(duplicate_rows)
```

```
Duplicate Rows:
24
```

As shown above, there is 24 duplicate rows in the data set.

24 duplicate row
After dropping the 24 duplicate Rows, new dataset size:
2087 rows x 17 columns

```
data=data.drop_duplicates()  # use dataframe.drop_duplicates() to drop the duplicate rows
display(data)
```

	Age	Gender	Height	Weight	CALC	FAVC	FCVC	NCP	SCC	SMOKE	CH2O	family_history_with_overweight
0	21.000000	Female	1.620000	64.000000	no	no	2.0	3.0	no	no	2.000000	yes
1	21.000000	Female	1.520000	56.000000	Sometimes	no	3.0	3.0	yes	yes	3.000000	yes
2	23.000000	Male	1.800000	77.000000	Frequently	no	2.0	3.0	no	no	2.000000	yes
3	27.000000	Male	1.800000	87.000000	Frequently	no	3.0	3.0	no	no	2.000000	no
4	22.000000	Male	1.780000	89.800000	Sometimes	no	2.0	1.0	no	no	2.000000	no
...	...	...	...	...	...	...	...	...	...	...	...	...
2106	20.976842	Female	1.710730	131.408528	Sometimes	yes	3.0	3.0	no	no	1.728139	yes
2107	21.982942	Female	1.748584	133.742943	Sometimes	yes	3.0	3.0	no	no	2.005130	yes
2108	22.524036	Female	1.752206	133.689352	Sometimes	yes	3.0	3.0	no	no	2.054193	yes
2109	24.361936	Female	1.739450	133.346641	Sometimes	yes	3.0	3.0	no	no	2.852339	yes
2110	23.664709	Female	1.738836	133.472641	Sometimes	yes	3.0	3.0	no	no	2.863513	yes

2087 rows × 17 columns

```
data.shape # check data set dimension after removing duplicates
```

```
(2087, 17)
```

NAN VALUES CHECK

```python
# Check if the set has NaN values
print(data.isna())
```

```
        Age  Gender  Height  Weight   CALC   FAVC   FCVC    NCP    SCC  SMOKE  \
0     False   False   False   False  False  False  False  False  False  False
1     False   False   False   False  False  False  False  False  False  False
2     False   False   False   False  False  False  False  False  False  False
3     False   False   False   False  False  False  False  False  False  False
4     False   False   False   False  False  False  False  False  False  False
...     ...     ...     ...     ...    ...    ...    ...    ...    ...    ...
2106  False   False   False   False  False  False  False  False  False  False
2107  False   False   False   False  False  False  False  False  False  False
2108  False   False   False   False  False  False  False  False  False  False
2109  False   False   False   False  False  False  False  False  False  False
2110  False   False   False   False  False  False  False  False  False  False

       CH2O  family_history_with_overweight    FAF    TUE   CAEC  MTRANS  \
0     False                           False  False  False  False   False
1     False                           False  False  False  False   False
2     False                           False  False  False  False   False
3     False                           False  False  False  False   False
4     False                           False  False  False  False   False
...     ...                             ...    ...    ...    ...     ...
2106  False                           False  False  False  False   False
2107  False                           False  False  False  False   False
2108  False                           False  False  False  False   False
2109  False                           False  False  False  False   False
2110  False                           False  False  False  False   False

      NObeyesdad
0          False
1          False
2          False
3          False
4          False
...          ...
2106       False
2107       False
2108       False
2109       False
2110       False

[2087 rows x 17 columns]
```

No NaN values in the dataset

INFINITY VALUES CHECK

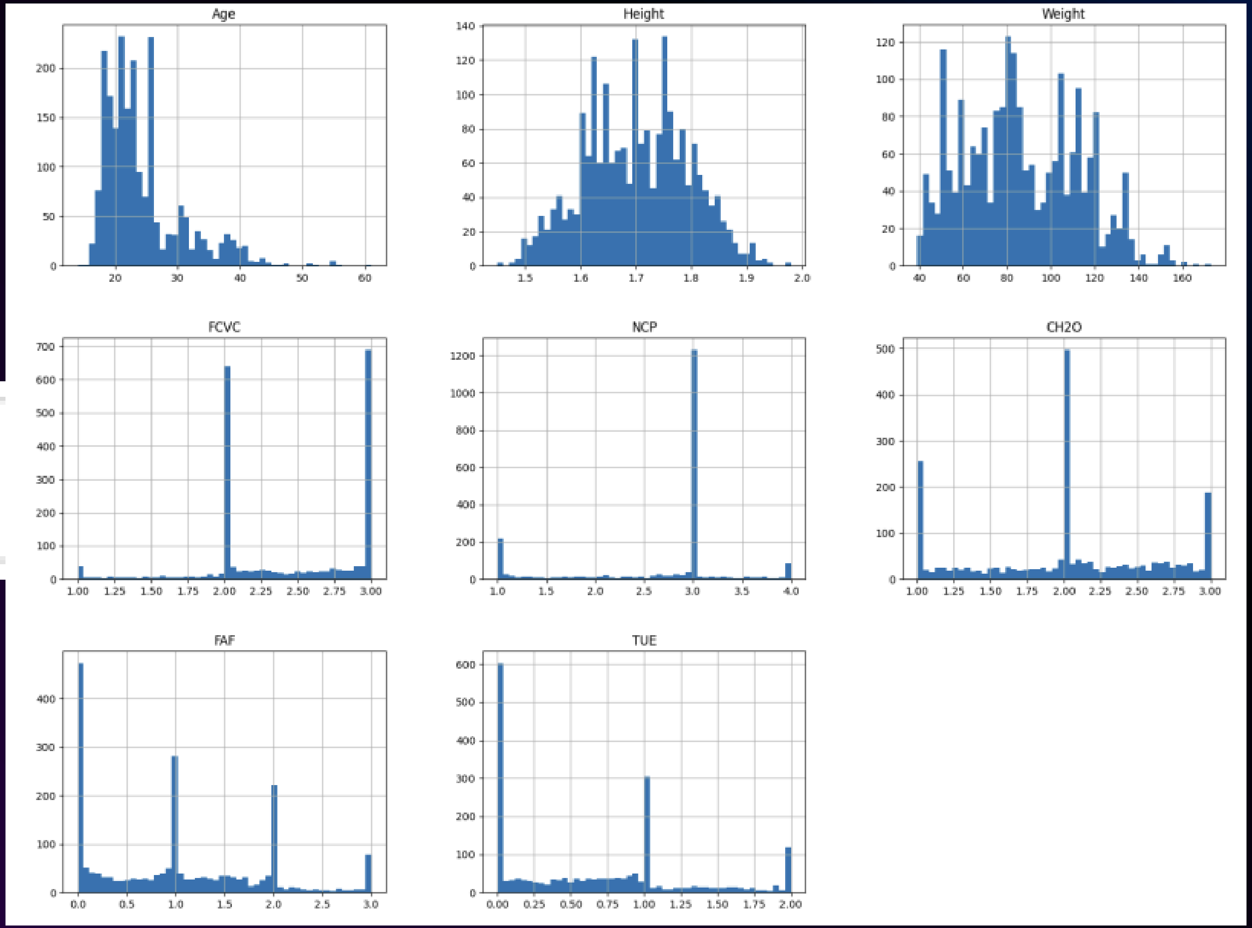
```python
# Check the presence of infinity values in the data set
data.isin([np.inf, -np.inf])
print(data)
```

```
            Age  Gender    Height      Weight        CALC FAVC  FCVC  NCP  \
0     21.000000  Female  1.620000   64.000000          no   no   2.0  3.0
1     21.000000  Female  1.520000   56.000000   Sometimes   no   3.0  3.0
2     23.000000    Male  1.800000   77.000000  Frequently   no   2.0  3.0
3     27.000000    Male  1.800000   87.000000  Frequently   no   3.0  3.0
4     22.000000    Male  1.780000   89.800000   Sometimes   no   2.0  1.0
...         ...     ...       ...         ...         ...  ...   ...  ...
2106  20.976842  Female  1.710730  131.408528   Sometimes  yes   3.0  3.0
2107  21.982942  Female  1.748584  133.742943   Sometimes  yes   3.0  3.0
2108  22.524036  Female  1.752206  133.689352   Sometimes  yes   3.0  3.0
2109  24.361936  Female  1.739450  133.346641   Sometimes  yes   3.0  3.0
2110  23.664709  Female  1.738836  133.472641   Sometimes  yes   3.0  3.0

      SCC SMOKE      CH2O family_history_with_overweight       FAF       TUE  \
0      no    no  2.000000                            yes  0.000000  1.000000
1     yes   yes  3.000000                            yes  3.000000  0.000000
2      no    no  2.000000                            yes  2.000000  1.000000
3      no    no  2.000000                             no  2.000000  0.000000
4      no    no  2.000000                             no  0.000000  0.000000
...   ...   ...       ...                            ...       ...       ...
2106   no    no  1.728139                            yes  1.676269  0.906247
2107   no    no  2.005130                            yes  1.341390  0.599270
2108   no    no  2.054193                            yes  1.414209  0.646288
2109   no    no  2.852339                            yes  1.139107  0.586035
2110   no    no  2.863513                            yes  1.026452  0.714137

           CAEC                 MTRANS           NObeyesdad
0     Sometimes  Public_Transportation        Normal_Weight
1     Sometimes  Public_Transportation        Normal_Weight
2     Sometimes  Public_Transportation        Normal_Weight
3     Sometimes                Walking   Overweight_Level_I
4     Sometimes  Public_Transportation  Overweight_Level_II
...         ...                    ...                  ...
2106  Sometimes  Public_Transportation     Obesity_Type_III
2107  Sometimes  Public_Transportation     Obesity_Type_III
2108  Sometimes  Public_Transportation     Obesity_Type_III
2109  Sometimes  Public_Transportation     Obesity_Type_III
2110  Sometimes  Public_Transportation     Obesity_Type_III

[2087 rows x 17 columns]
```

No $\pm\infty$ values in the data set

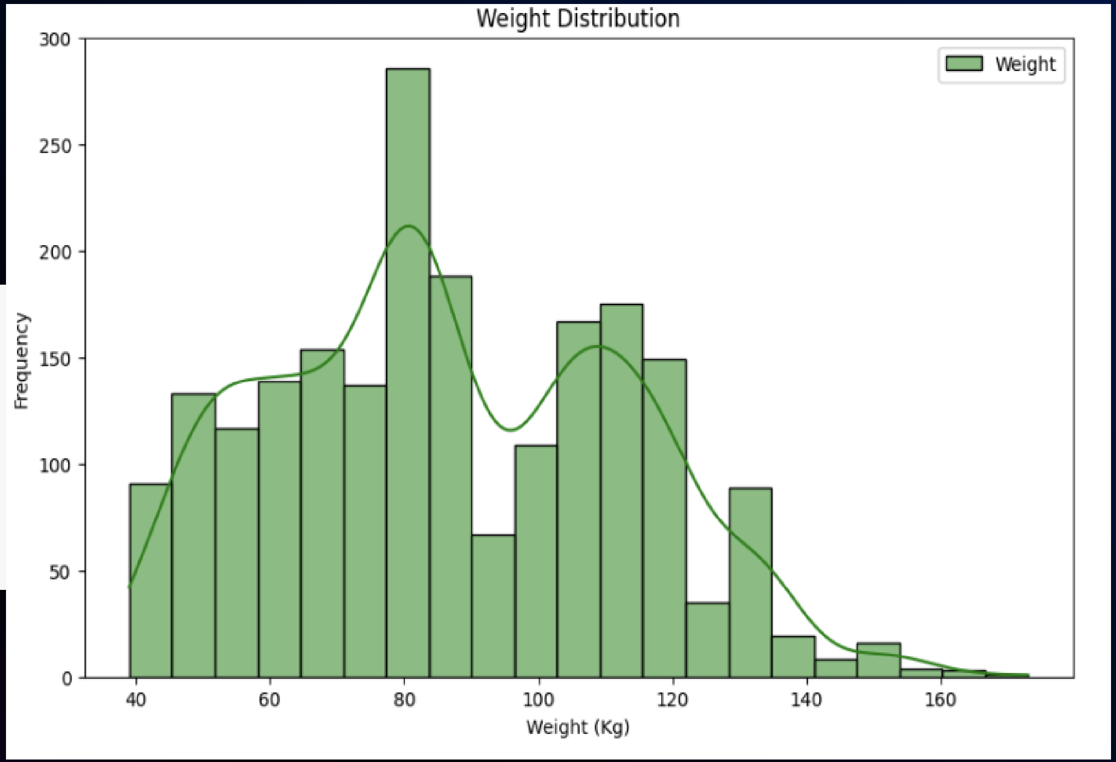
GENERATING HISTOGRAM PLOTS FOR THE DATASET ATTRIBUTES

```python
# Plot histograms for different parameters
data.hist(bins=50, figsize=(20,15));
```



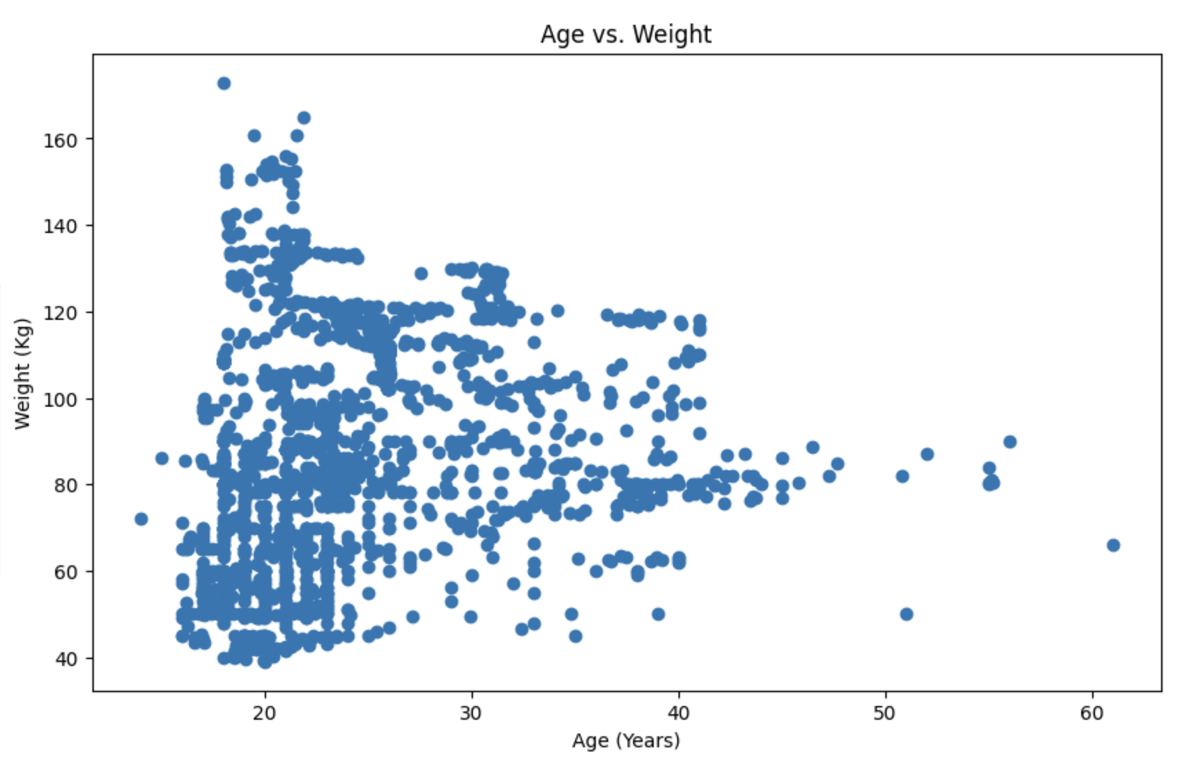
HISTOGRAM PLOT SHOWING THE WEIGHT DISTRIBUTION

```
# Plot Weight Distribution
plt.figure(figsize=(10, 6))
sns.histplot(data=data, x='Weight', color='green', kde=True, label='Weight')
plt.title('Weight Distribution')
plt.xlabel('Weight (Kg)')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```



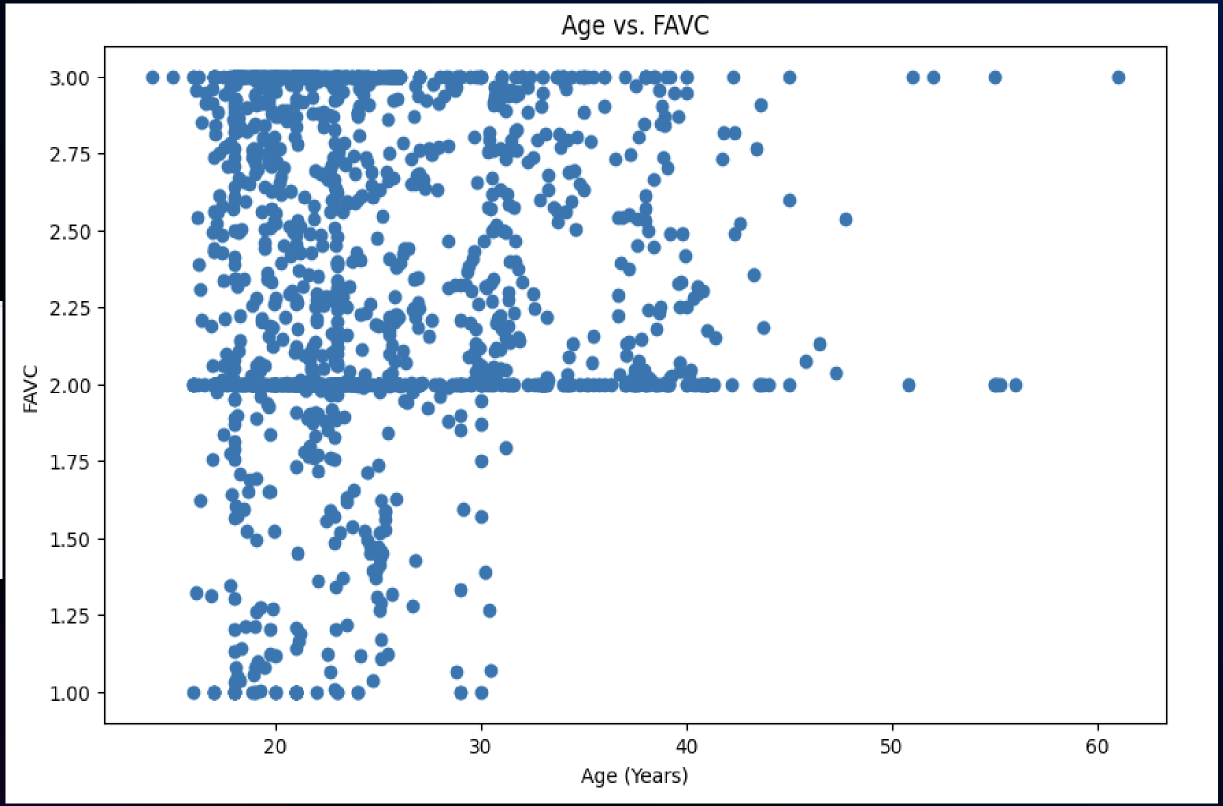
SCATTER PLOT DEPICTING AGE VS. WEIGHT

```
fig, ax = plt.subplots(figsize=(10,6))
ax.scatter(data['Age'], data['Weight'])
plt.title('Age vs. Weight')
ax.set_xlabel('Age (Years)')
ax.set_ylabel('Weight (Kg)')
plt.show()
```



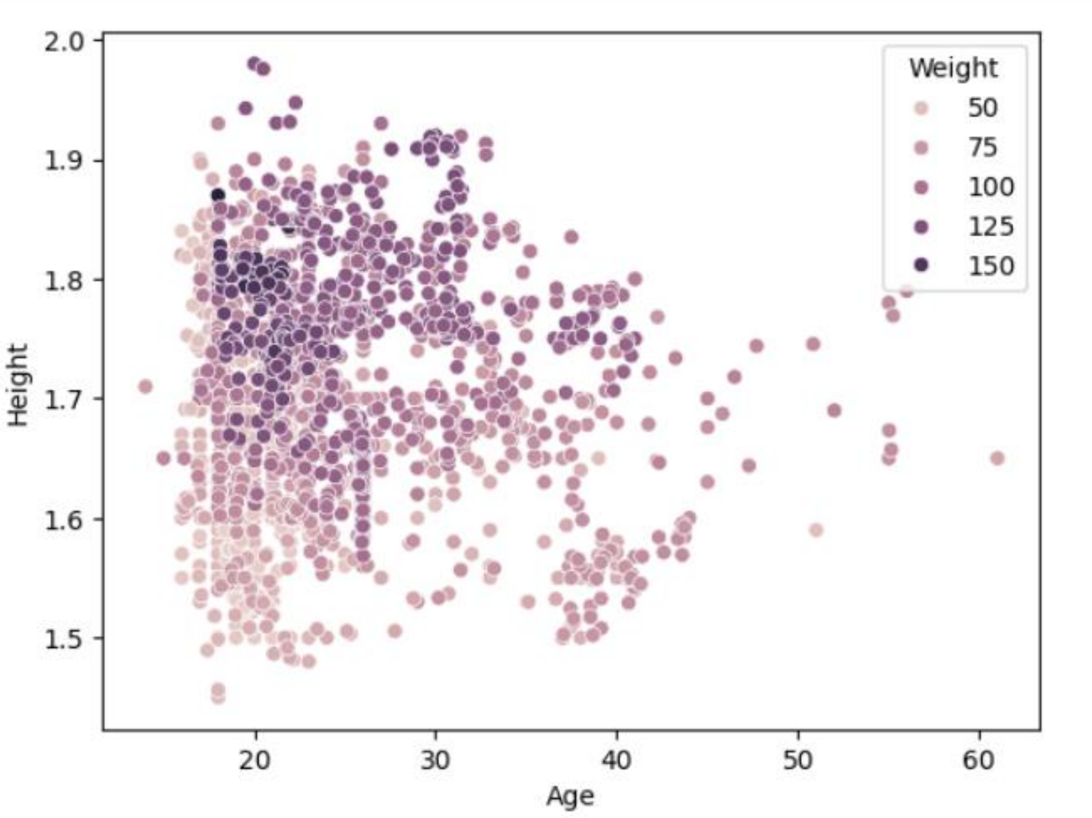
SCATTER PLOT SHOWING AGE VS. FCVC

```python
fig, ax = plt.subplots(figsize=(10,6))
ax.scatter(data['Age'], data['FCVC'])
plt.title('Age vs. FAVC')
ax.set_xlabel('Age (Years)')
ax.set_ylabel('FAVC')
plt.show()
```



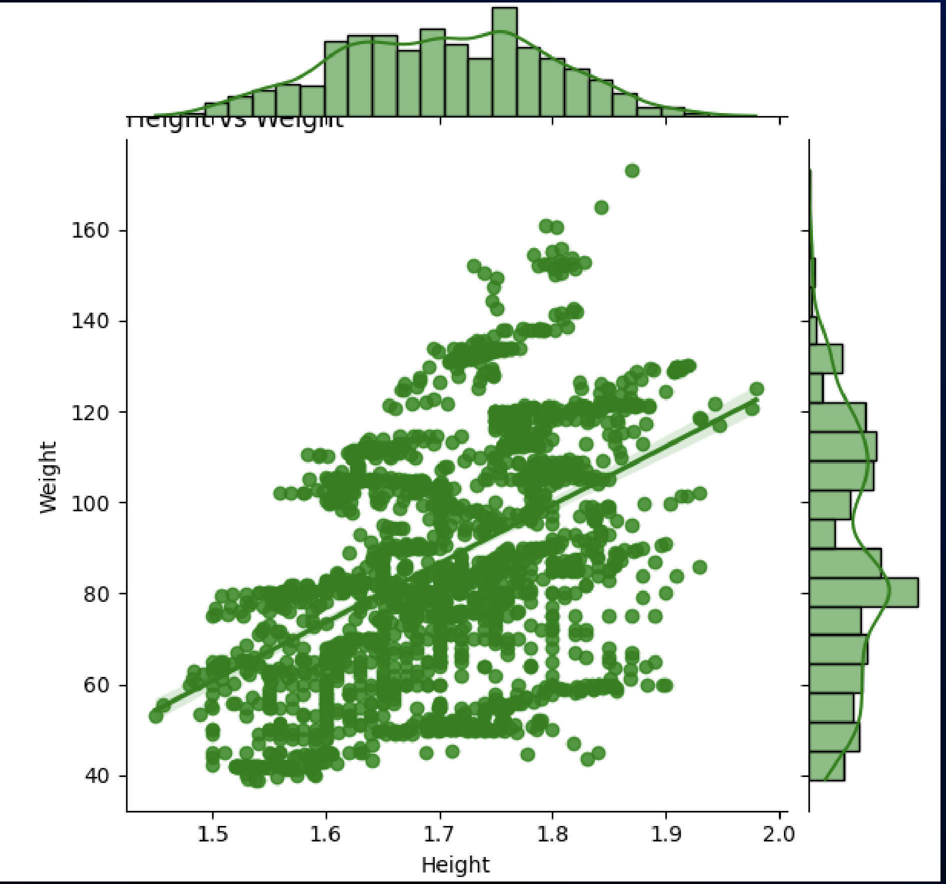
SCATTER PLOT ILLUSTRATING AGE VS. HEIGHT FOR DIFFERENT WEIGHTS

```
scatter_P =sns.scatterplot(x= "Age" ,y= "Height",
                           hue="Weight",
                           data=data);
```

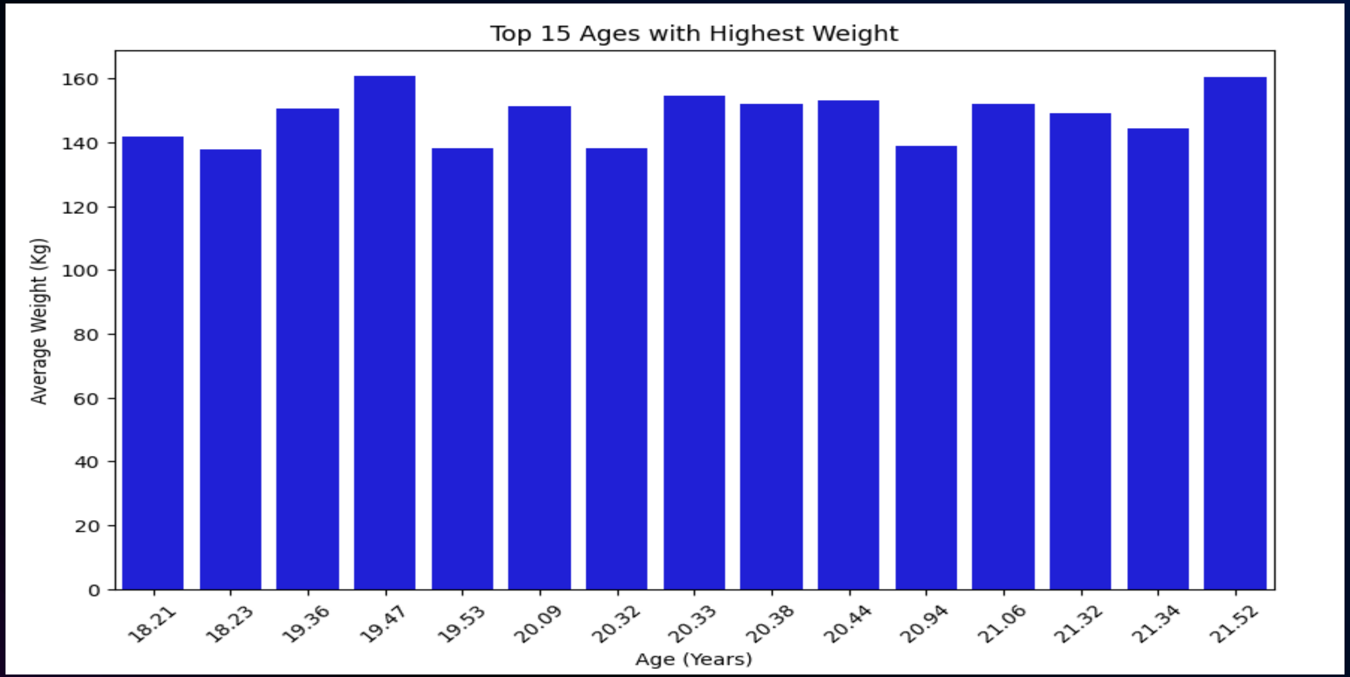


USING 'JOINTPLOT' FOR PLOTTING HEIGHT VS. WEIGHT

```python
sns.jointplot(x="Height",y='Weight',data=data,kind='reg', color='green')
plt.title("Height vs Weight",loc='left')
```



BAR PLOT SHOWING AGE VS. AVERAGE WEIGHT FOR THE TOP 15 AGES

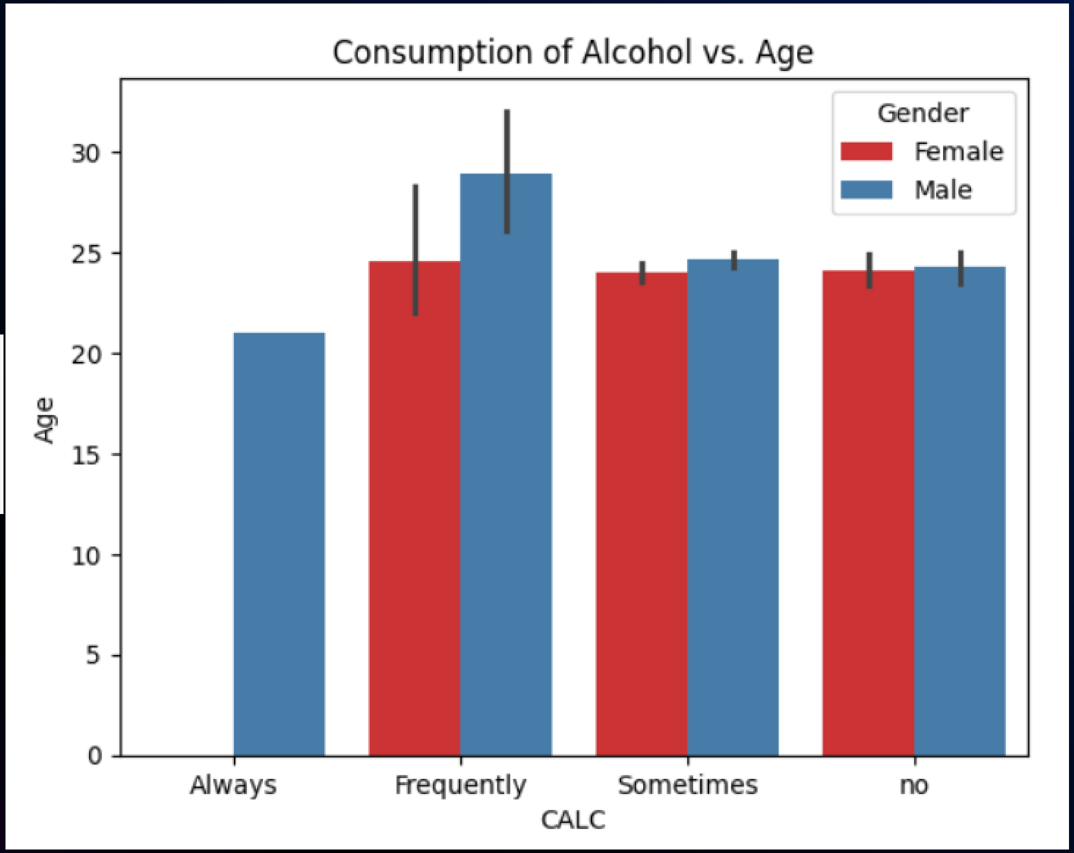



```python
# Top 15 Ages with Highest Weight
top_15_ages = data.groupby('Age')['Weight'].mean().nlargest(15)
plt.figure(figsize=(10, 6))
sns.barplot(x=top_15_ages.index, y=top_15_ages.values, color='blue')
plt.title('Top 15 Ages with Highest Weight')
plt.xlabel('Age (Years)')
plt.ylabel('Average Weight (Kg)')
plt.xticks(rotation=45)
plt.show()
```

```python
data['Age']=data['Age'].round(2)
```

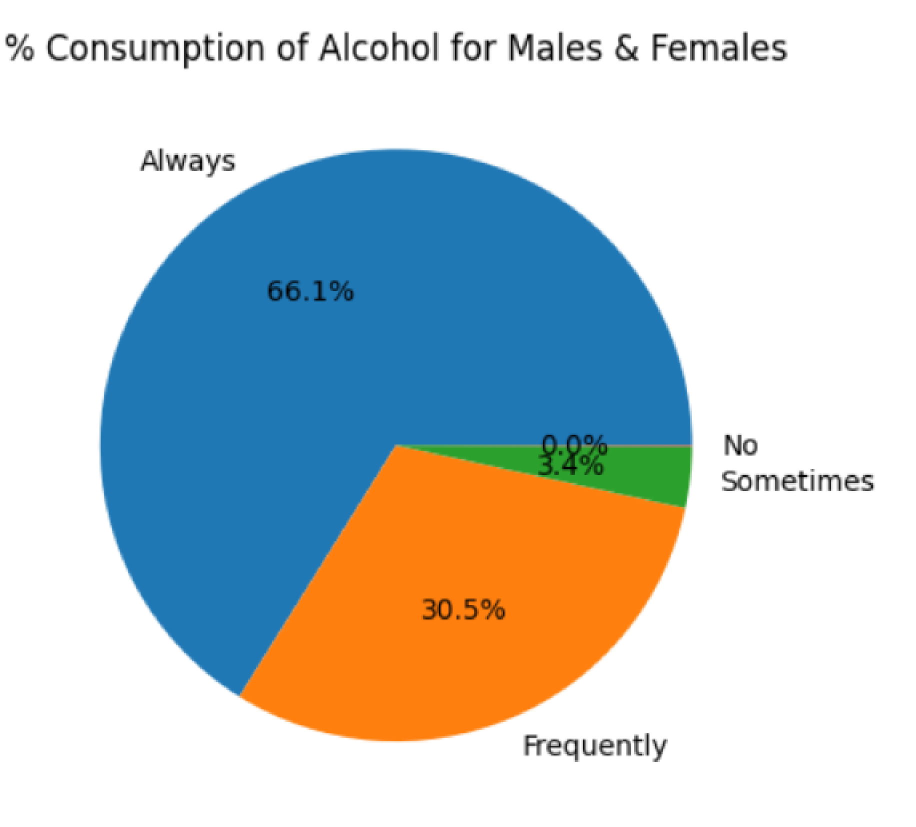
BARPLOT ILLUSTRATING CONSUMPTION OF ALCOHOL FOR FEMALES & MALES

```python
data['CALC'] = data['CALC'].astype('category')
sns.barplot(x='CALC', y='Age',data=data,hue='Gender', palette='Set1')
plt.title('Consumption of Alcohol vs. Age')
plt.show()
```



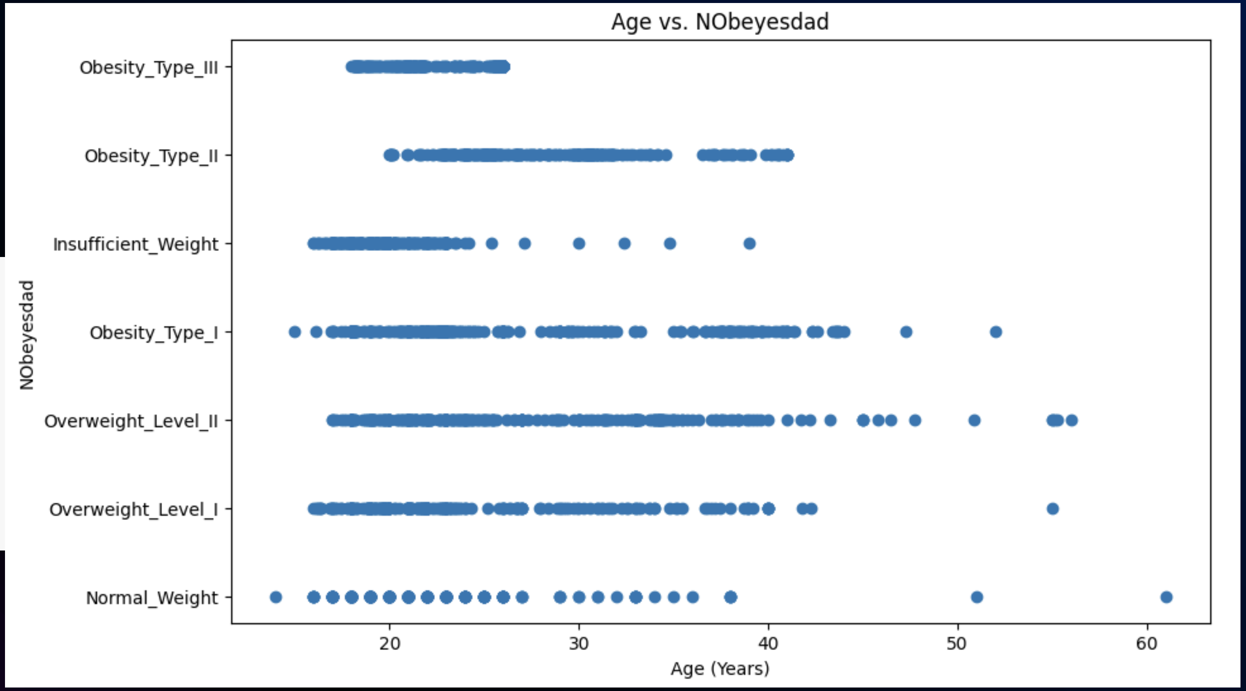
PIE CHART SHOWING THE % CONSUMPTION OF ALCOHOL

```
count = data['CALC'].value_counts()
labels = ["Always", "Frequently", "Sometimes", "No"]
vals = count.values
plt.pie(vals, labels=labels, autopct="%1.1f%%")
plt.title("% Consumption of Alcohol for Males & Females")

plt.show()
```



PLOTTING AGE VS. NOBEYESAD USING SCATTER PLOT

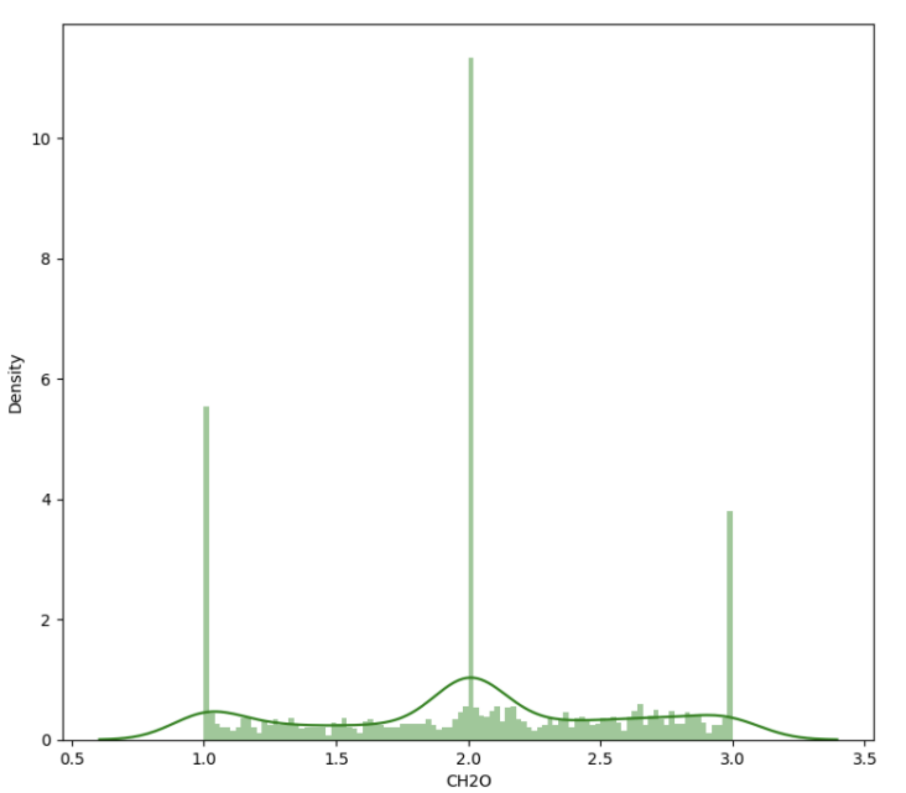
```
# Age vs. NObeyesdad
fig, ax = plt.subplots(figsize=(10,6))
ax.scatter(data['Age'], data['NObeyesdad'])
plt.title('Age vs. NObeyesdad')
ax.set_xlabel('Age (Years)')
ax.set_ylabel('NObeyesdad')
plt.show()
```



PLOTTING THE DISTRIBUTION OF CH2O

```python
print(data['CH2O'].describe())
plt.figure(figsize=(9, 8))
sns.distplot(data['CH2O'], color='g', bins=100, hist_kws={'alpha': 0.4});
```

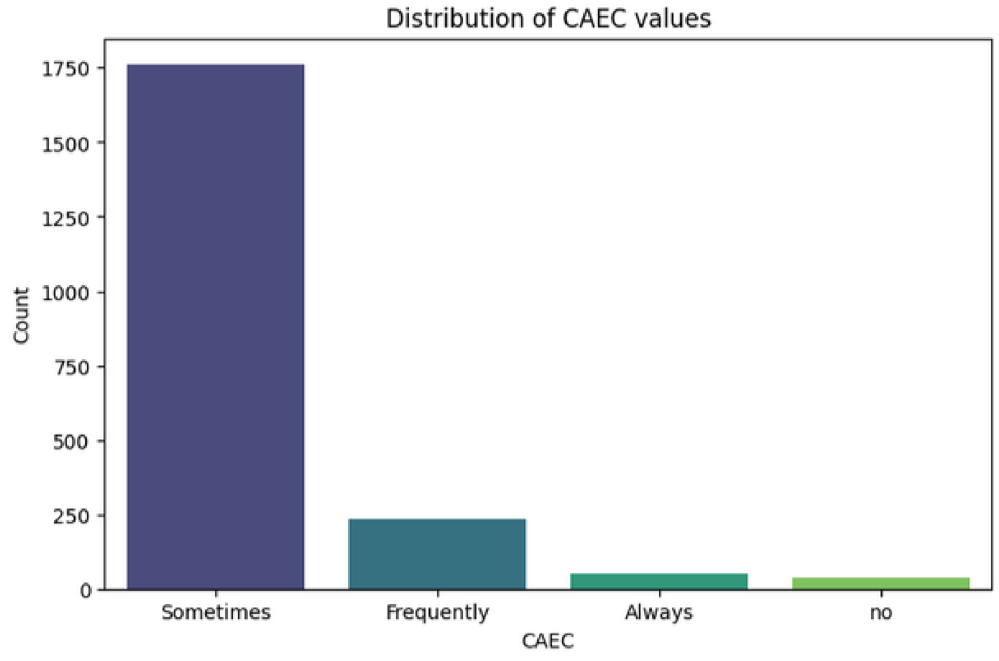
```
count    2087.000000
mean        2.004749
std         0.608284
min         1.000000
25%         1.590922
50%         2.000000
75%         2.466193
max         3.000000
Name: CH2O, dtype: float64
```



https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

COUNT PLOT SHOWING THE CATEGORICAL FACTOR 'CAEC'

```python
# Distribution of CAEC values
plt.figure(figsize=(8, 5))
sns.countplot(data=data, x='CAEC', palette='viridis')
plt.title('Distribution of CAEC values')
plt.xlabel('CAEC')
plt.ylabel('Count')
plt.show()
```

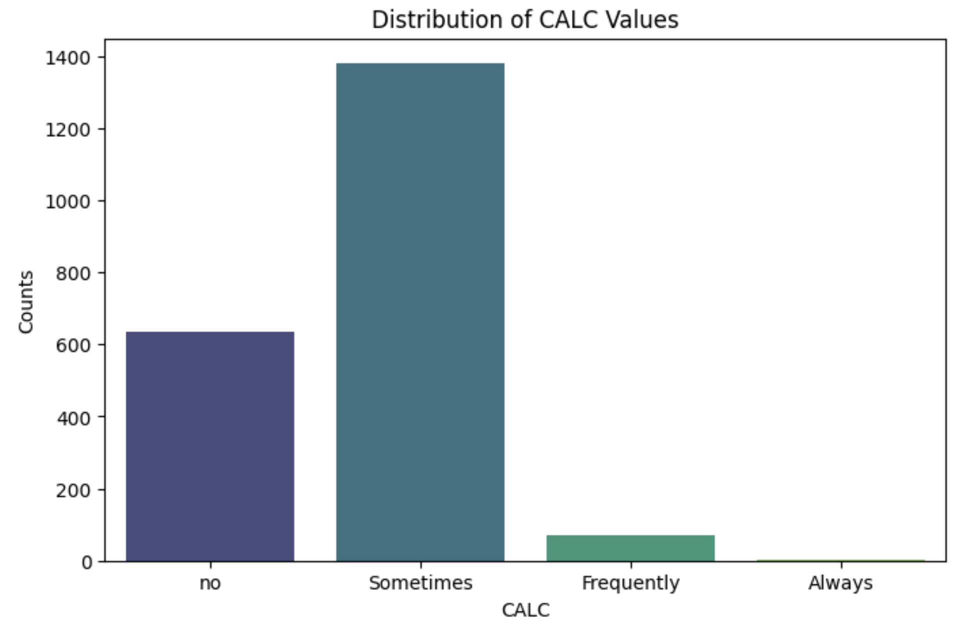


COUNT PLOT DEPICTING THE CATEGORICAL FACTOR 'CALC'

```python
data['CALC'].unique()
```

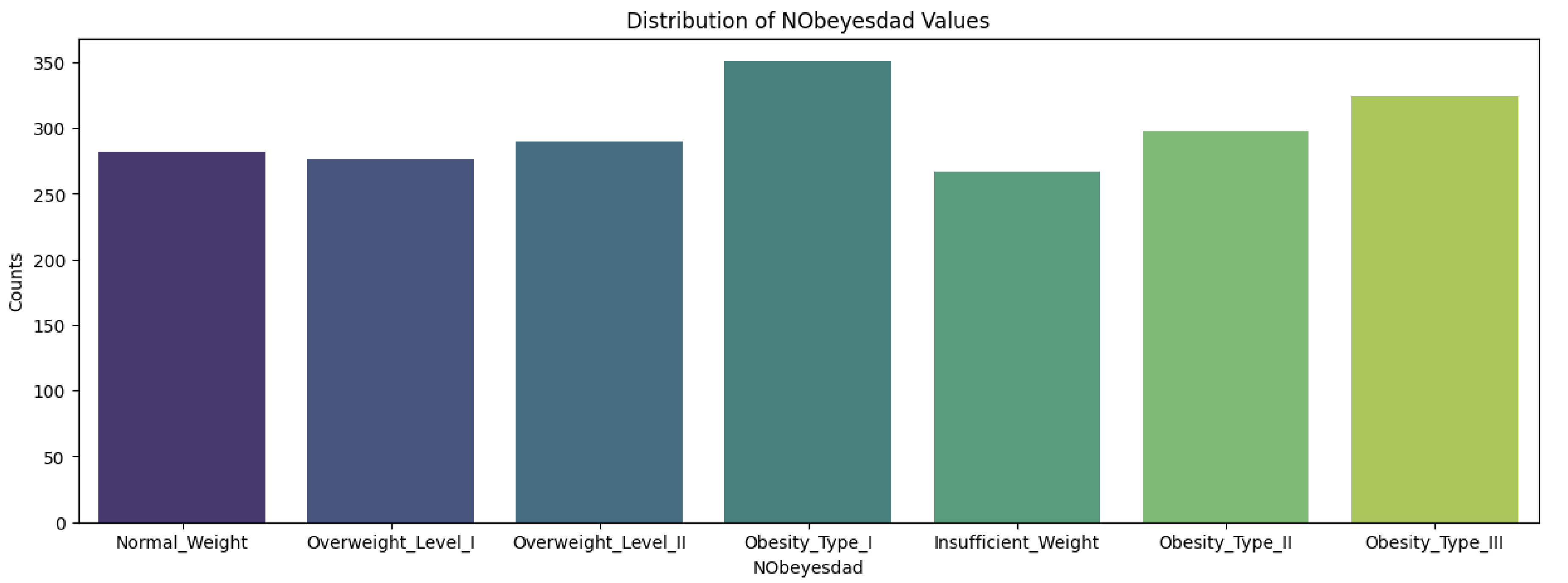
```
array(['no', 'Sometimes', 'Frequently', 'Always'], dtype=object)
```

```python
# CALC distribution
plt.figure(figsize=(8,5))
sns.countplot(data=data, x='CALC', palette='viridis')
plt.title('Distribution of CALC Values')
plt.xlabel ('CALC')
plt.ylabel('Counts')
plt.show()
```



COUNT PLOT DEPICTING NOBEYESDAD VS. FREQUENCY

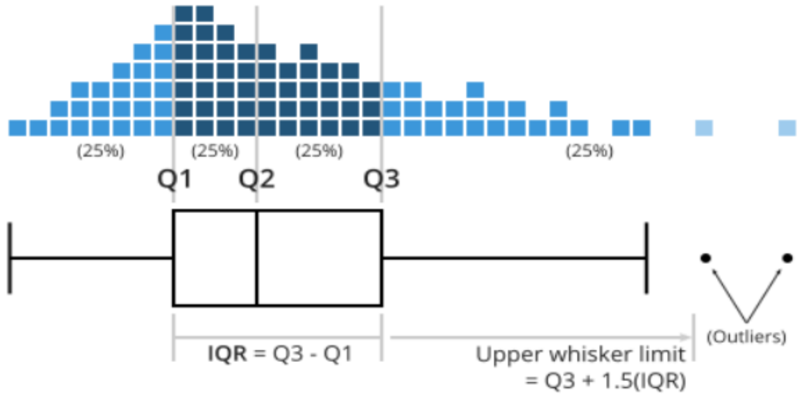
```python
# NObeyesdad distribution
plt.figure(figsize=(15,5))
sns.countplot(data=data, x='NObeyesdad', palette='viridis')
plt.title('Distribution of NObeyesdad Values')
plt.xlabel ('NObeyesdad')
plt.ylabel('Counts')
plt.show()
```



HOW TO INTERPRETING A BOX PLOT?

Interpreting a box and whiskers

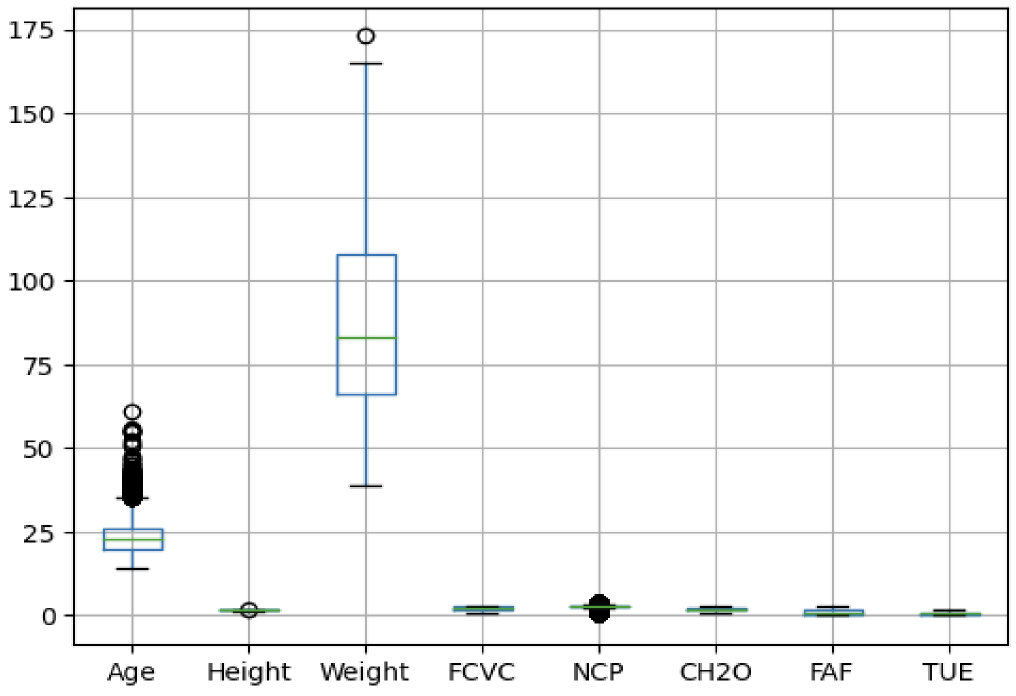
Construction of a box plot is based around a dataset's quartiles, or the values that divide the dataset into equal fourths. The first quartile (Q1) is greater than 25% of the data and less than the other 75%. The second quartile (Q2) sits in the middle, dividing the data in half. Q2 is also known as the median. The third quartile (Q3) is larger than 75% of the data, and smaller than the remaining 25%. In a box and whiskers plot, the ends of the box and its center line mark the locations of these three quartiles.



https://www.atlassian.com/data/charts/box-plot-complete-guide

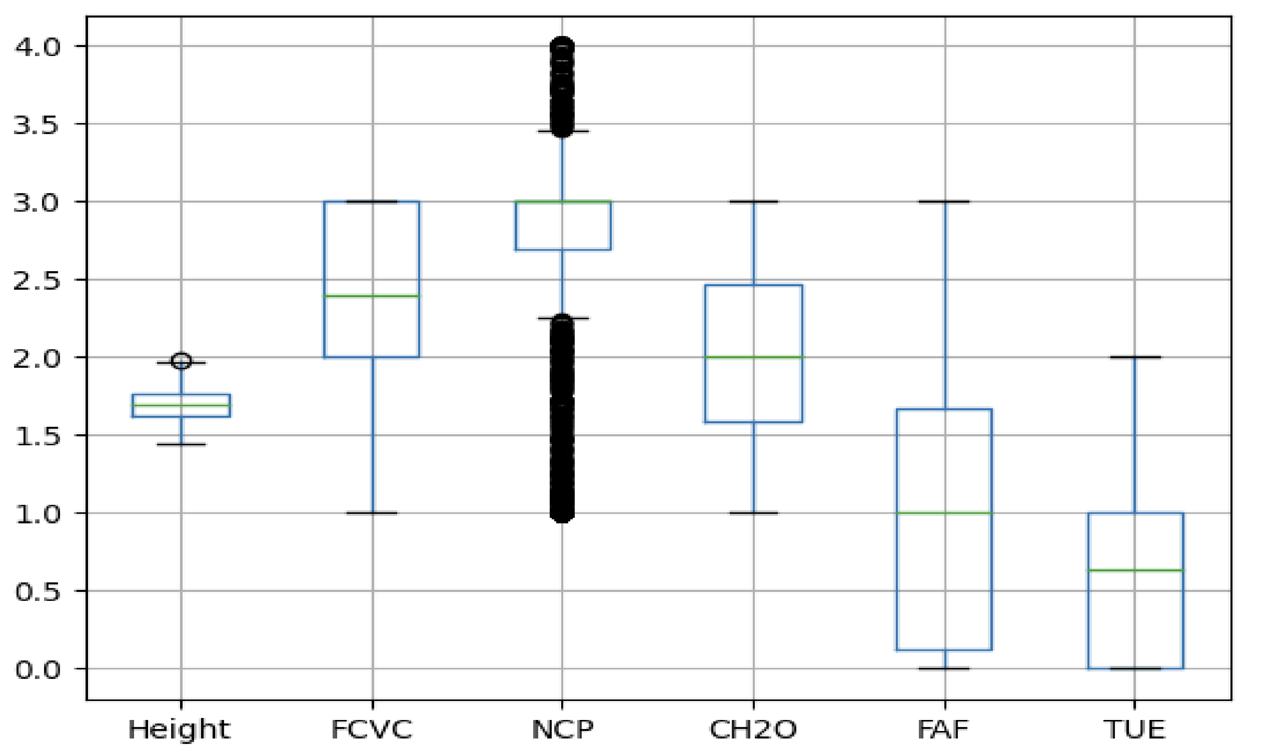
INSPECTING THE DATASET FOR OUTLIERS USING BOXPLOT

```
columns=['Age', 'Height', 'Weight','FCVC', 'NCP','CH2O','FAF','TUE' ]
boxplot = data.boxplot(column=['Age', 'Height', 'Weight','FCVC', 'NCP','CH2O','FAF','TUE'])
```



INSPECTING THE DATASET FOR OUTLIERS USING BOXPLOT

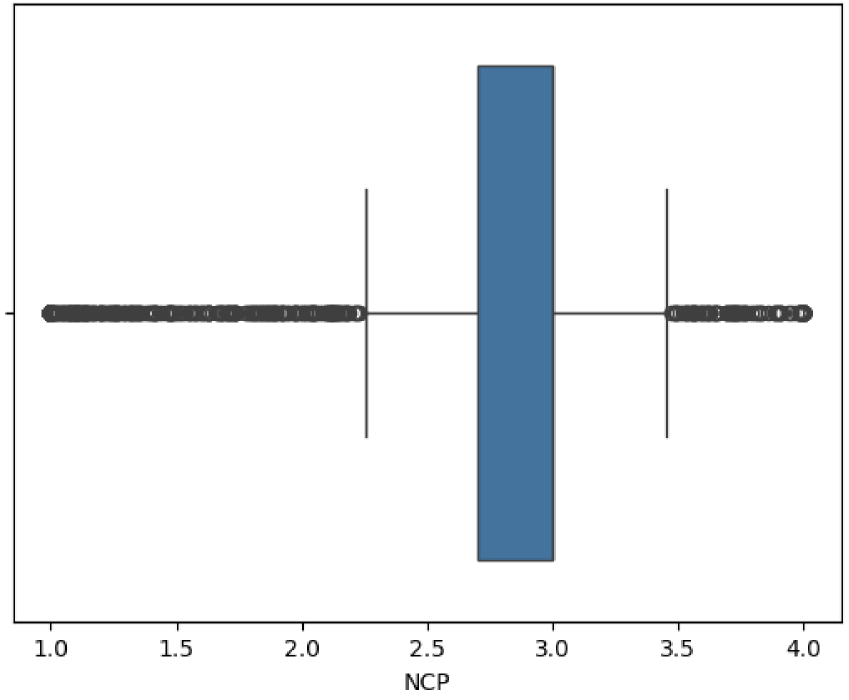
```
columns=['Height', 'Weight','FCVC', 'NCP','CH2O','FAF','TUE' ]
boxplot = data.boxplot(column=['Height','FCVC', 'NCP','CH2O','FAF','TUE'])
```



BOXPLOT SHOWING THE NCP FACTOR VALUES

```
1 sns.boxplot(x=data['NCP'])
```

```
<Axes: xlabel='NCP'>
```



THE DATA SET FACTORS BEFORE REMOVING THE OUTLIERS

```
[16]
# define path
data_path = '/content/sample_data/diabetes.csv'

# import/load data into a newly created dataframe, df
df = pd.read_csv(data_path)
```

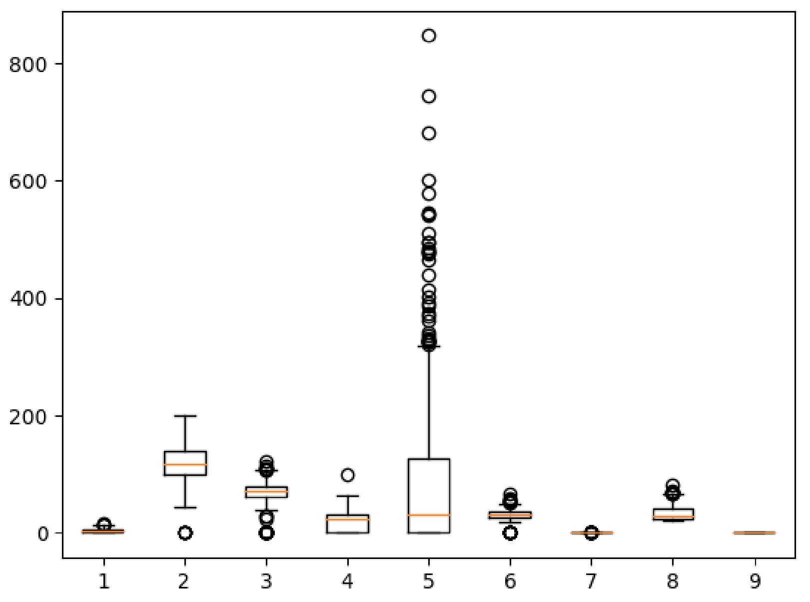
```
[17]
df.columns
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

Data Set Attribute Information:

- Pregnancies
- Glucose
- BloodPressure
- SkinThickness
- Insulin
- BMI
- Diabetes Pedigree Function
- Age
- Outcome

```
plt.boxplot(df);
```



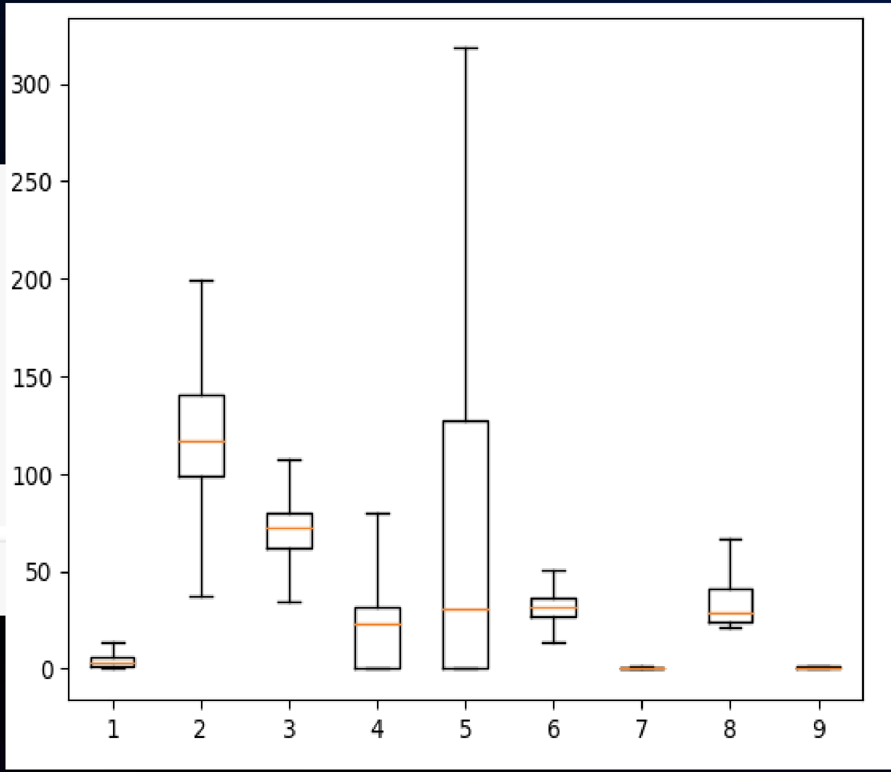
Example showing the removal of outliers using the diabetes.csv dataset
https://www.kaggle.com/datasets/saurabh00007/diabetescsv

THE DATASET FACTORS AFTER REMOVING THE OUTLIERS

[37]

```
# Treat outliers in the data set
def outlier_removal():
    l = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
        'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
    for i in l:
        x = np.quantile(df[i],[0.25,0.75])
        iqr = x[1]-x[0]
        upper = x[1]+1.5*iqr
        lower = x[0]-1.5*iqr
        df[i] = np.where(df[i]>upper,upper,(np.where(df[i]<lower,lower,df[i])))

outlier_removal()
```

```
# Plot data after removing outliers
plt.boxplot(df);
```



https://www.geeksforgeeks.org/detect-and-remove-the-outliers-using-python/

WHAT IS A CORRELATION?

Correlation is a statistical indicator that quantifies the degree to which two variables change in relation to each other. It indicates the strength and direction of the linear relationship between two variables. The correlation coefficient is denoted by “r”, and it ranges from -1 to 1.

- If r = -1, it means that there is a perfect negative correlation.
- If r = 0, it means that there is no correlation between the two variables.
- If r = 1, it means that there is a perfect positive correlation.

There are two popular methods used to find the correlation coefficients:

Pearson’s product-moment correlation coefficient

The Pearson correlation coefficien t (r) is a measure of linear relationship between two variables.

$$r = n(\sum xy) – (\sum x)(\sum y) / \sqrt{}[n \sum x^2 - (\sum x)^2][n \sum y^2 - (\sum y)^2]$$

Here,

- n is the number of data points
- $\sum xy$ is the sum of the product of corresponding values of x and y
- $\sum x$ is the sum of all the values of x
- $\sum y$ is the sum of all the values of y
- $\sum x^2$ is the sum of the squares of all values of x
- $\sum y^2$ is the sum of the squares of all the of y

https://www.geeksforgeeks.org/create-a-correlation-matrix-using-python

WHAT IS A CORRELATION MATRIX?

A correlation is a tabular representation that displays correlation coefficients, indicating the strength and direction of relationships between variables in a dataset. Within this matrix, each cell signifies the correlation between two specific variables. This tool serves multiple purposes, serving as a summary of data relationships, input for more sophisticated analyses, and a diagnostic aid for advanced analytical procedures. By presenting a comprehensive overview of inter-variable correlations, the matrix becomes invaluable in discerning patterns, guiding further analyses, and identifying potential areas of interest or concern in the dataset. Its applications extend beyond mere summary statistics, positioning it as a fundamental component in the preliminary stages of diverse and intricate data analyses.

https://www.geeksforgeeks.org/create-a-correlation-matrix-using-python/

INTERPRETING THE CORRELATION MATRIX RESULTS

Strong correlations, indicated by values close to 1 or -1, suggest a robust connection, while weak correlations, near 0, imply a less pronounced association. They are identifying these degrees of correlation aids in understanding the intensity of interactions within the dataset, facilitating targeted analysis and decision-making. Positive correlations (values > 0) signify that as one variable increases, the other tends to increase as well. Conversely, negative correlations (values < 0) imply an inverse relationship—when one variable increases, the other tends to decrease. Investigating these directional associations provides insights into how variables influence each other, crucial for formulating informed hypotheses and predictions.

https://www.geeksforgeeks.org/create-a-correlation-matrix-using-python/

CALCULATING THE PAIRWISE CORRELATION FOR ALL COLUMNS

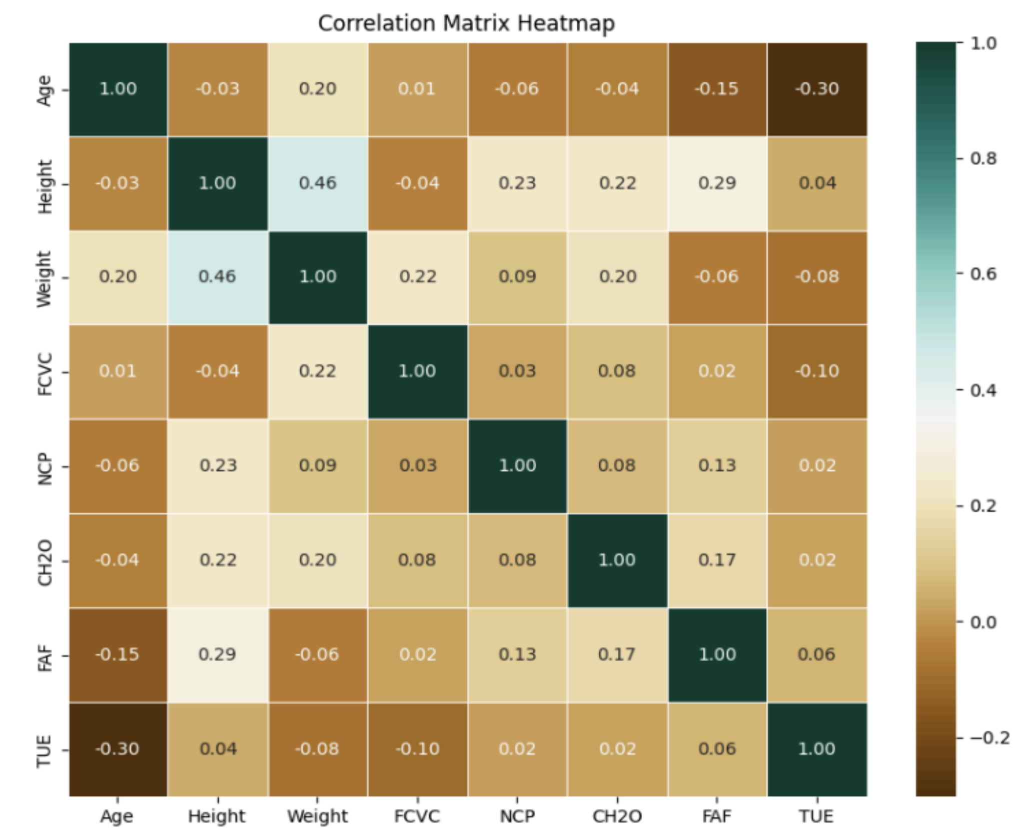
```python
numeric_df = data.select_dtypes(include=['number'])# Select only numeric columns
# Calculate correlation matrix
correlation_matrix = numeric_df.corr()
# Print the correlation matrix
print("Correlation Matrix:")
print(correlation_matrix)
```

```
Correlation Matrix:
             Age    Height    Weight      FCVC       NCP      CH2O       FAF  \
Age     1.000000 -0.031748  0.198160  0.013572 -0.055823 -0.044058 -0.148202
Height -0.031748  1.000000  0.457468 -0.040363  0.227806  0.220487  0.293584
Weight  0.198160  0.457468  1.000000  0.216574  0.092149  0.203823 -0.056490
FCVC    0.013572 -0.040363  0.216574  1.000000  0.034885  0.081332  0.022003
NCP    -0.055823  0.227806  0.092149  0.034885  1.000000  0.075335  0.127816
CH2O   -0.044058  0.220487  0.203823  0.081332  0.075335  1.000000  0.165310
FAF    -0.148202  0.293584 -0.056490  0.022003  0.127816  0.165310  1.000000
TUE    -0.302927  0.041808 -0.079351 -0.104128  0.015693  0.020704  0.058716

             TUE
Age    -0.302927
Height  0.041808
Weight -0.079351
FCVC   -0.104128
NCP     0.015693
CH2O    0.020704
FAF     0.058716
TUE     1.000000
```

https://www.geeksforgeeks.org/python-pandas-dataframe-corr/

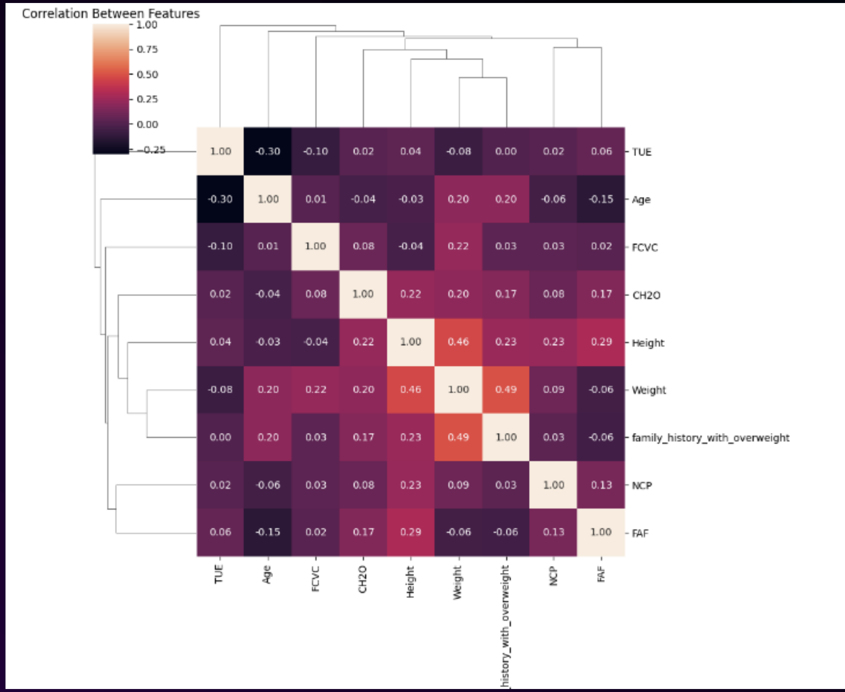
PLOTTING THE CORRELATION MATRIX HEATMAP

```python
# Print the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='BrBG', fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```



https://www.geeksforgeeks.org/create-a-correlation-matrix-using-python/

CORRELATION MATRIX CLUSTER MAP




```python
numeric_df = data.select_dtypes(include=['number'])# Select numeric columns only
# Calculate correlation matrix
correlation_matrix = numeric_df.corr()
sns.clustermap(correlation_matrix, annot=True, fmt=".2f")
plt.title("Correlation Between Features")
```

CONVERTING CATEGORICAL VARIABLES TO NUMERIC VALUES

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC
0	Female	21.000000	1.620000	64.000000	yes	no
1	Female	21.000000	1.520000	56.000000	yes	no
2	Male	23.000000	1.800000	77.000000	yes	no
3	Male	27.000000	1.800000	87.000000	no	no
4	Male	22.000000	1.780000	89.800000	no	no

Before

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP
0	Female	21.0	1.62	64.0	1	0	2.0	3.0
1	Female	21.0	1.52	56.0	1	0	3.0	3.0
2	Male	23.0	1.80	77.0	1	0	2.0	3.0
3	Male	27.0	1.80	87.0	0	0	3.0	3.0
4	Male	22.0	1.78	89.8	0	0	2.0	1.0

After

```
# replacing values
data['family_history_with_overweight'].replace(['no', 'yes'],
                                                [0, 1], inplace=True)
data.head(5)
```

Converting categorical variables to numerical values for use in machine learning predictive models

https://www.geeksforgeeks.org/how-to-convert-categorical-variable-to-numeric-in-pandas/

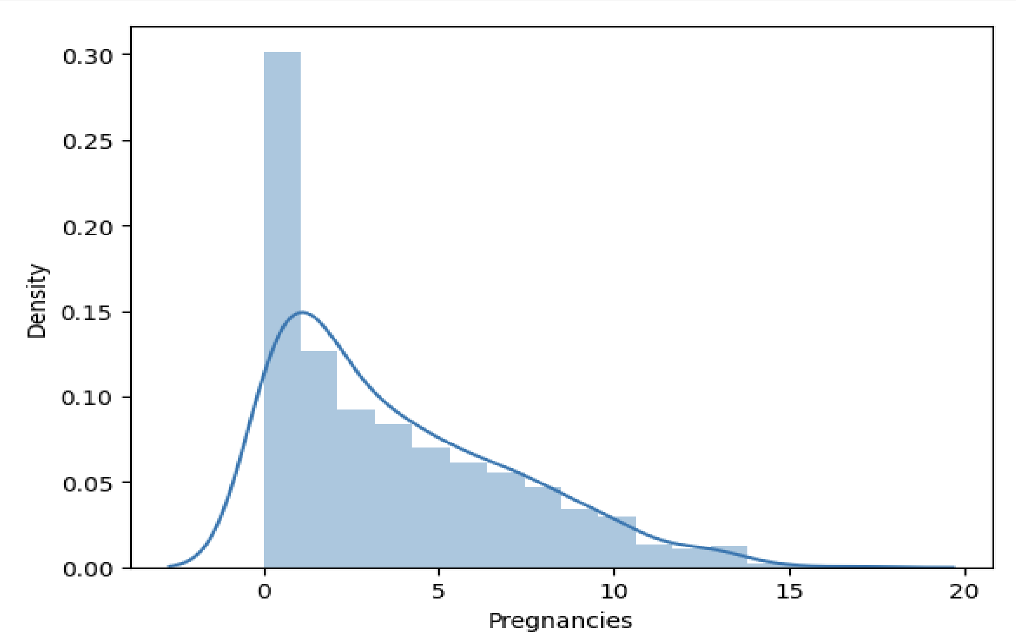
CONCLUSIONS

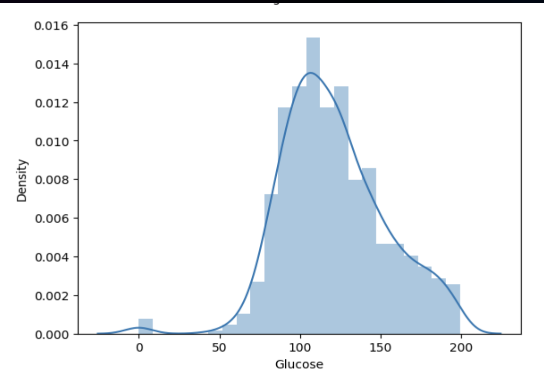
- **The original dataset contains 2111 rows and 17 column**
- **There are 24 duplicate rows in the dataset**
- **All values in the dataset are unique and contain no null, NaN, $\pm\infty$ or missing values**
- **The NCP (number of main meals per day) data contains some outlier values and required removal**
- **There is a significant correlation between weight and height**

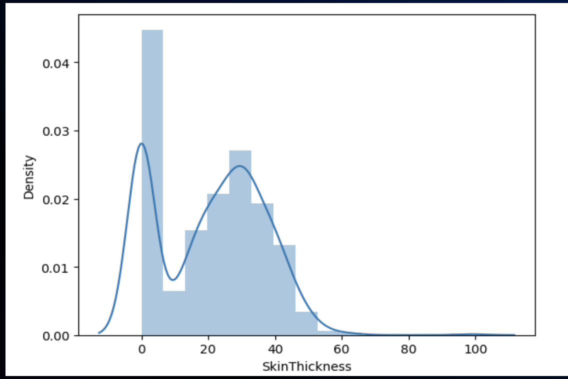
ADDITIONAL EXAMPLES

DISTRIBUTION PLOT ILLUSTRATION

```python
data_feature = data.columns

for feature in data_feature:
    p = sns.distplot(a = data[feature])
    plt.show()
```







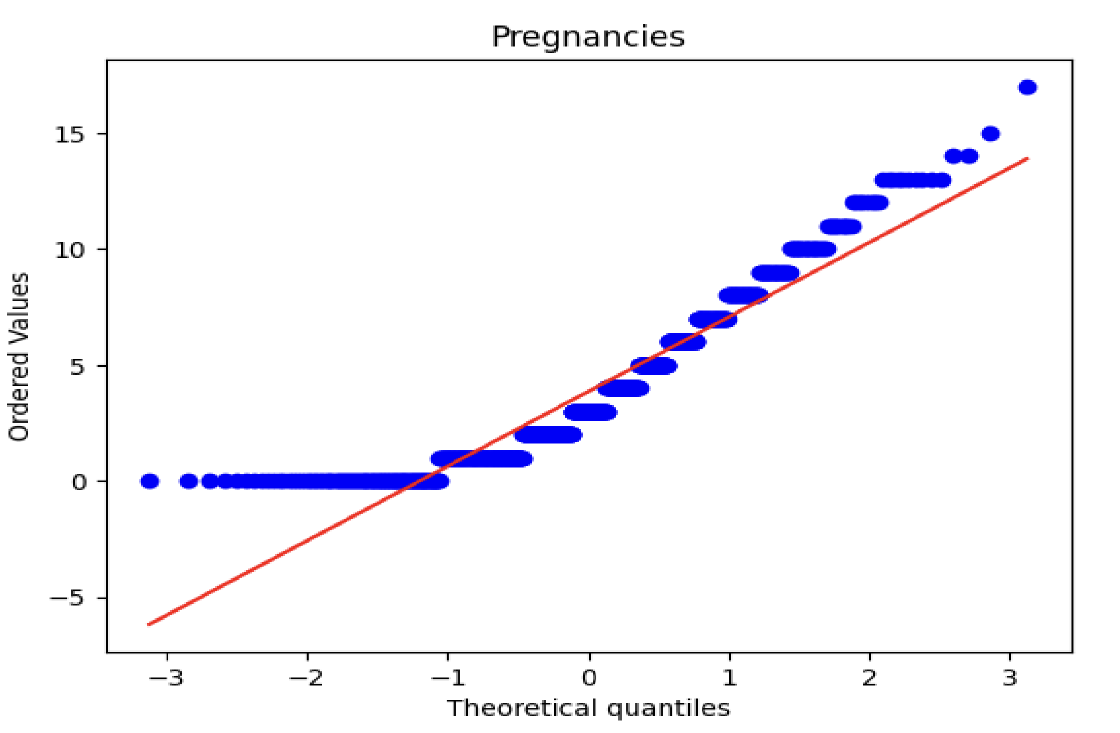

```python
data = pd.read_csv("/content/sample_data/diabetes.csv") # Load data set using pandas
data.head(3)
```

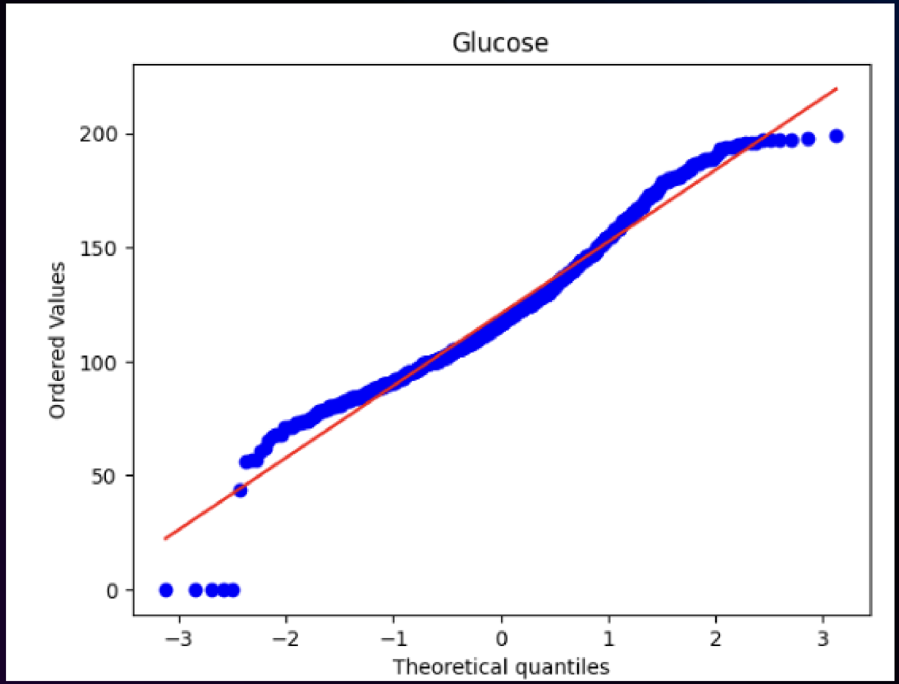
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1

https://www.kaggle.com/datasets/mathchi/diabetes-data-set

PROBABILITY PLOT FOR TWO FACTORS IN THE DATASET

```python
import scipy
from scipy import stats
for feature in data.columns:
    stats.probplot(data[feature], plot = plt)
    plt.title(feature)
    plt.show()
```

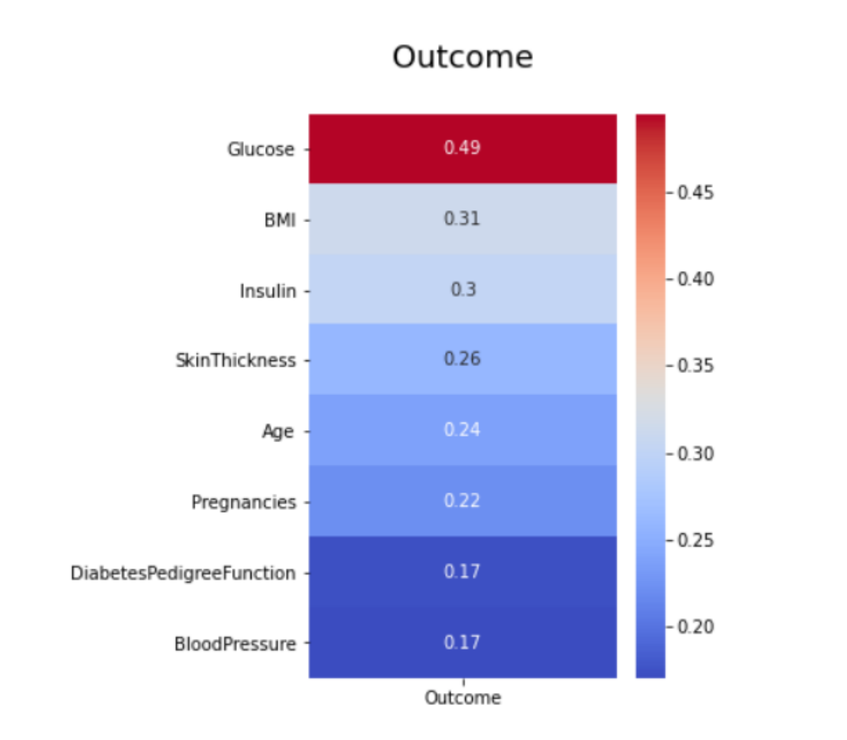




https://www.kaggle.com/datasets/mathchi/diabetes-data-set

ANOTHER FORM OF HEATMAP PRESENTATION

```
def corr_to_target(dataframe, target, title=None, file=None):
    plt.figure(figsize=(4,6))
    sns.heatmap(dataframe.corr()[[target]].sort_values(target,
                                                        ascending=False)
[1:],
                                                        annot=True,
                                                        cmap='coolwarm')

    plt.title(f'\n{title}\n', fontsize=18)

    plt.show();

    return

corr_to_target(df, "Outcome", title="Outcome")
```



https://www.kaggle.com/code/busekseolu/diabetes-classification

REFERENCES

1. **Data Science Horizon, Data cleaning and preprocessing for data science beginners.** (https://datasciencehorizons.com/data-cleaning-preprocessing-data-science-beginners-ebook/)
2. **Matthieu Komorowski, et al., Exploratory Data Analysis, Chapter 15, doi:10.1007/978-3- 319-43742-2_15.**
3. **DataQuest, Data Science Cheat Sheet-Pandas.** (https://s3.amazonaws.com/dq-blog-files/pandas-cheat-sheet.pdf)
4. **DataCamp, Python for Data Science Cheat Sheet-Matplotlib.** (https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Matplotlib_Cheat_Sheet.pdf)
5. **DataQuest, Data Science Cheat Sheet-Numpy.** (https://assets.datacamp.com/blog_assets/Numpy_Python_Cheat_Sheet.pdf)
6. **DataCamp, Python for Data Science, Seaborn Cheat Sheet.** (https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Seaborn_Cheat_Sheet.pdf)
7. **DataCamp, Python for Data Science Cheat Sheet-Scikit-Learn.** (https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Scikit_Learn_Cheat_Sheet_Python.pdf)
8. **David Beazley, et al., Python Cookbook, 3rd Ed., O'Reilly, Beiing, 2013.**